Minimización del *makespan* para el problema de máquinas paralelas no relacionadas con tiempos de *setup* dependientes de la secuencia mediante un algoritmo híbrido VNS/ACO*

*Eduardo Salazar Hornig***
*Gina Soto Gavilán****

Recibido: 29/10/2018 • Aceptado: 22/10/2020
https://doi.org/10.22395/rium.v20n38a11

Resumen

Se propone una heurística híbrida combinando *Variable Neighborhood Search* (VNS) y *Ant Colony Optimization* (ACO) para resolver el problema de programación de máquinas paralelas no relacionadas con tiempos de preparación dependientes de la secuencia con el objetivo de minimizar el *makespan*. La búsqueda en entornos variables se propone con un esquema descendente resolviendo en una primera etapa el problema de programación de los trabajos a las máquinas, y luego, en una segunda etapa, un algoritmo ACO, reordena sucesivamente los trabajos en la máquina de mayor *makespan*. Se realizan pruebas experimentales sobre un conjunto de problemas de prueba de la literatura, mostrando que al aplicar la segunda etapa de la metaheurística propuesta se mejoran las soluciones obtenidas en la primera etapa del algoritmo y que al comparar los resultados obtenidos con otros métodos de la literatura resulta ser un método competitivo.

Palabras clave: *makespan*, máquinas paralelas no relacionadas, tiempos de preparación dependientes de la secuencia, metaheurística, VNS, ACO.

* Artículo de investigación.

** Doctor en Investigación de Operaciones. Profesor Asociado, Departamento de Ingeniería Industrial, Universidad de Concepción, Chile. Correo electrónico: esalazar@udec.cl. Orcid: 0000-0003-4387-0877

*** Magíster en Ingeniería Industrial, Universidad de Concepción, Chile. Correo electrónico: ginasotog@udec.cl

Makespan Minimization on Unrelated Parallel Machines Scheduling Problem with Sequence Dependent Setup Times by a VNS/ACO Hybrid Algorithm

Abstract

This paper proposes a hybrid heuristic that combines Variable Neighborhood Search (VNS) with Ant Colony Optimization (ACO) to solve the scheduling problem of nonrelated parallel machines with sequence dependent setup times in order to minimize the makespan. The Variable Neighborhood Search is proposed to solve the scheduling problem with a descending scheme in a first phase, with an ACO algorithm, which successively reorder the jobs in the machine with the largest makespan in a second phase. An experimental study was performed using test problems from the literature showing that the second phase of the algorithm improves the solution obtained in the first phase. The results obtained are also compared with other methods in the literature proving to be a competitive method.

Keywords: makespan, non related parallel machines, setup, metaheuristic, VNS, ACO.

INTRODUCCIÓN

El problema de máquinas paralelas no relacionadas con tiempos de preparación (*setup*) dependientes de la secuencia ha suscitado el interés de investigadores por sus aplicaciones en la industria textil, química, de la pintura, del plástico y del papel, entre otras. Para establecer una relación de confianza con sus clientes las organizaciones requieren sistematizar su proceso de programación y control de la producción que les permita mejorar su productividad. Una eficiente programación de la producción representa uno de los mayores desafíos actuales para una organización, especialmente cuando se tienen tiempos de *setup* dependientes de la secuencia. En este trabajo, el objetivo de la secuenciación es la minimización del *makespan* o tiempo de finalización del último trabajo procesado.

Debido a la naturaleza NP-hard de este problema (ver [1]), se requiere de algoritmos heurísticos que entreguen buenas soluciones (cercanas a la óptima) en tiempos computacionales razonables. Entre las investigaciones relevantes para el problema de estudio, están los trabajos [1], en el que se propone un algoritmo *Tabu Search* (TS) que utiliza dos fases de perturbación; en [2] los autores proponen Meta-RaPS, un algoritmo aleatorizado de multi-partida, y en [3] se propone ACO II, un algoritmo *Ant Colony Optimization* (ACO) mejorado de dos fases. En [4] el problema se resuelve a través de un algoritmo genético (GA) que incorpora dos búsquedas locales, en [5] se propone una mejora de un algoritmo *Simulated Annealing* (SA) incorporando una estrategia de búsqueda restringida que elimina movimientos no efectivos y en [6] se presenta una propuesta de solución a través de la combinación de ACO, SA y *Variable Neighborhood Search* (VNS) en la que se aprovechan las mejores características de cada uno de los métodos para mejorarla solución. La programación de máquinas paralelas es un problema conocido en la práctica, en el que por lo general se presentan particularidades de la operación del sistema, como lo es que la existencia de máquinas no relacionadas lleve a un problema de elegibilidad de máquinas [5] o deba considerarse la reprogramación debido a paradas de máquinas en el horizonte de programación [6]. En [7] los autores desarrollancuatro heurísticas y un algoritmo de mejora para estimar la frontera de Pareto en la optimización bi-objetivo del *makespan* y una penalidad, mientras que en [8] los autores proponen un método de reasignación de trabajos utilizando *Simulated Annealing* (con parámetros determinados por un controlador de lógica difusa) para optimizar la función objetivo que combina *makespan*, estabilidad de máquinas y un costo de penalización.

1. FORMULACIÓN DEL PROBLEMA

El problema de máquinas paralelas no relacionadas con tiempos de setup y minimización de *makespan*, denotado por $R_m \mid s_{ijk} \mid C_{max}$, consiste en programar sin interrupción,

n trabajos independientes que se encuentran disponibles para su proceso en $t = 0$, en m máquinas en paralelo de manera que el procesamiento de todos los trabajos se realice en el menor tiempo posible. Además, se considera que los tiempos de proceso p_{jk} del trabajo j en la máquina k varía de una máquina a otra, cuyas velocidades de proceso no guardan una relación proporcional. Con respecto a los trabajos, éstos solo son procesados por una máquina y cada máquina solo puede procesar un trabajo a la vez. La principal particularidad que presenta este problema es la consideración de tiempos de *setup* dependientes de la secuencia, lo que se refiere a que el tiempo de preparación que requiere una máquina k para procesar un trabajo j depende del trabajo previamente procesado por la máquina. Es decir, el orden con que se procesan los trabajos en cada máquina resulta relevante. Así, s_{ijk} denota el tiempo de setup requerido para procesar el trabajo j a continuación del trabajo i en la máquina k. El objetivo de la programación es la minimización del makespan, $C_{max} = max_{j=1, \ldots, n} \{C_j\}$ (con C_j fecha de finalización del trabajo j), que corresponde al intervalo de tiempo transcurrido entre $t = 0$ y la mayor fecha de finalización de un trabajo procesado. Los supuestos del problema son:

- El número de máquinas es menor al número de trabajos ($m < n$).
- Todas las máquinas están disponibles en $t = 0$.
- Todos los trabajos están liberados en $t = 0$ y tienen igual importancia.
- Los trabajos son procesados solo en una máquina y sin interrupción.
- Las máquinas operan sin fallas y procesan un trabajo a la vez.
- Los trabajos son independientes (no tienen relaciones de precedencia).
- El inicio del proceso del primer trabajo en cada máquina es $t = 0$.

En este trabajo el problema $R_m \mid s_{ijk} \mid C_{max}$ se resuelve mediante un algoritmo híbrido VNS/ACO, comparándolo con al algoritmo VNS propuesto en este trabajo y otras heurísticas de la literatura. Para la comparación de los algoritmos se utilizó un conjunto de instancias de prueba de la literatura [1].

La metaheurística VNS fue propuesta por [9] y se basa en la idea de realizar cambios sistemáticos de la vecindad de búsqueda, con la idea de escapar de óptimos locales y obtener así una mejor solución. El desempeño del esquema VNS dependerá de la calidad de la solución inicial, técnicas de búsqueda local, vecindades utilizadas, y el orden en que éstas se aplican [10]. La simpleza de su estructura y elementos facilitan las decisiones de modificación que mejoran la heurística y la obtención de mejores resultados [11]. ACO fue introducida por [12] basada en el comportamiento

de una colonia de hormigas que al desplazarse desde la fuente de alimento a su nido optimizan su ruta en forma colectiva.

Para ilustrar el problema, se considera un sistema de dos máquinas paralelas no relacionadas y seis trabajos a programar [13]. Los parámetros de este problema se presentan en las tablas 1 y 2. Los valores de la diagonal en la tabla 2 representan los tiempos de *setup* iniciales, es decir, el tiempo de preparación si el trabajo asociado a la posición de la diagonal es el primer trabajo que se procesa en la respectiva máquina.

Tabla 1. Tiempos de proceso de los trabajos (p_{jk})

Trabajo	1	2	3	4	5	6
p_{j1}	9	12	7	15	10	6
p_{j2}	11	8	10	11	13	8

Fuente: [13].

Tabla 2. Matrices de tiempos de *setup* (s_{ijk})

S_{ij1}	1	2	3	4	5	6
1	3	8	6	7	2	4
2	9	6	5	4	2	7
3	6	7	4	8	4	9
4	7	8	7	4	6	6
5	8	4	3	9	2	3
6	6	5	4	9	3	2
S_{ij2}	**1**	**2**	**3**	**4**	**5**	**6**
1	2	7	8	5	6	8
2	4	8	4	8	7	5
3	8	6	5	9	7	3
4	3	4	6	7	5	4
5	4	9	8	5	9	7
6	9	3	7	7	8	6

Fuente: [13].

La figura 1 ilustra una programación del problema especificado en las tablas 1 y 2 (en estas tablas se muestran en subrayado los tiempos de proceso y *setup* involucrados). En M1 se procesan los trabajos 6, 3 y 5 (en ese orden) finalizando su proceso en $t = 33$, mientras que en M2 se procesan los trabajos 1, 2 y 4 (en ese orden) finalizando su proceso en $t = 47$. Así, el makespan para la programación de la figura 1 es $C_{max} = 47$.

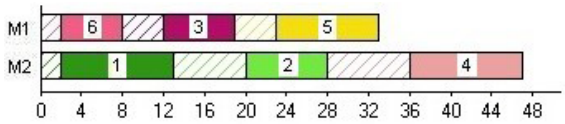

Figura 1. Carta Gantt – Programación

Fuente: [13].

2. ALGORITMO HÍBRIDO VNS/ACO

El algoritmo VNS/ACO propuesto se clasifica como una hibridización del tipo HRH (*high-level relay hybrid*) de las metaheurísticas VNS y ACO, en la que cada metaheurística trabaja de forma independiente, autónoma, sin cooperación y ejecutadas en forma secuencial. La solución entregada por VNS se utiliza como la solución de entrada de ACO, la que reordena sucesivamente los trabajos en la máquina de mayor *makespan* [14].

2.1 *Variable Neighborhood Search*

Según la vecindad utilizada por VNS, las soluciones se representan en forma secuencial o matricial. En la representación secuencial (secuencia de trabajos), cada trabajo es asignado sucesivamente a la máquina donde finaliza antes, mientras que en la representación matricial cada fila representa una máquina y las columnas asociadas a una fila (máquina) representan a la secuencia de trabajos a procesar por la respectiva máquina. Como mecanismo para el paso de una representación matricial a secuencial se construye una secuencia de trabajos ordenados de menor a mayor según su tiempo de finalización. La figura 2 ilustra gráficamente la estructura secuencial y matricial de una solución para un problema de $n = 9$ trabajos y $m = 3$ máquinas.

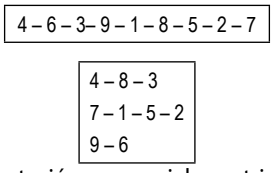

Figura 2. Representación secuencial y matricial de soluciones

Fuente: elaboración propia.

2.2 Vecindad *swap-secuencial* (*swap-s*): utiliza una estructura secuencial, y se construye mediante intercambios aleatorios de trabajos (no necesariamente consecutivos) un número controlado de veces.

2.3 Vecindad *swap-matricial* (*swap-m*): utiliza una estructura matricial y se construye mediante intercambios aleatorios de dos trabajos seleccionados en forma aleatoria en dos máquinas distintas, también seleccionadas aleatoriamente, un número controlado de veces. Luego se aplica la heurística MV del mejor vecino [15], que reasigna los trabajos en las máquinas seleccionadas para el intercambio buscando reducir el tiempo total de proceso en las máquinas donde se han modificado los trabajos.

procedure*setupECT()*
U = {1, 2, ... , n}
for *k=1, ... , m* ***do*** $M_k = 0$ ***endfor***
while $U \neq \emptyset$ ***do***
$t_{j(k)}=arg[\ min\ _{k=1,\ \dots\ ,\ m}\ \{\ M_k + p_{jk} + s_{ijk}\ \}];\ j \in U$
$j_0(k_0) \leftarrow arg[min\ _{j \in U}\ \{\ t_{j(k)}\ \}]$
Asignar trabajo j_0 *a máquina* k_0
$M_{k_0} \leftarrow M_{k_0} + p_{j_0 k_0} + s_{ij_0 k_0}$
$U \leftarrow U - \{j_0\}$
endwhile
endprocedure

Figura 3. Pseudo código heurística setupECT

Fuente: [13].

Para generar una solución inicial se utiliza la heurística *setup*ECT (ver figura 3) que incorpora los tiempos de *setup* a la heurística ECT original [16], asignando sucesivamente el trabajo que finaliza antes entre todas las máquinas (la programación que se ilustra en la figura 1 se obtiene utilizando la heurística *setupECT*).

En este estudio se utiliza la variante descendente de VNS, denotada por VND (variable neighborhood descent), la que parte de una solución inicial y se reemplaza iterativamente por una mejor solución entregada por el entorno o vecindad (ver figura 4). En la figura 4, *f* representa la medida de desempeño (función objetivo) a minimizar (en este caso corresponde al *makespan*) y la función *BúsquedaVecindad*(N_k (*s*)) realiza una búsqueda controlada en la vecindad N_k (*s*) (vecindad k de la solución *s*) entregando la solución *s'*, la mejor solución encontrada en la vecindad analizada.

procedure*VND()*
Seleccionar vecindades N_k, $k=1,\dots,k_{max}$
$s \leftarrow$ *SolucionInicial*
$k \leftarrow 1$
while $k \leq k_{max}$ ***do***
$s' \leftarrow BúsquedaVecindad(N_k(s))$
if $(f(s') < f(s))$ $\{s \leftarrow s';\ k \leftarrow 1\}$ *else* $\{k \leftarrow k+1\}$
endwhile
return *s*
endprocedure

Figura 4. Pseudocódigo del algoritmo VND

Fuente: elaboración propia.

Se utilizan dos vecindades ($k_{max} = 2$), *swap-s* y *swap-m*. Las figuras 5 y 6 ilustran cómo opera la generación de elementos cercanos a una solución para ambas vecindades [14]. El paso de la vecindad *swap-s* a *swap-m* se realiza asignando los trabajos en el orden dado por la secuencia a la máquina donde finaliza antes, mientras que para el paso de la vecindad *swap-m* a *swap-s* se define la secuencia de trabajos según C_j creciente.

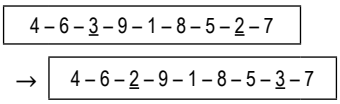

Figura 5. Intercambio Aleatorio (Secuencial)

Fuente: elaboración propia.

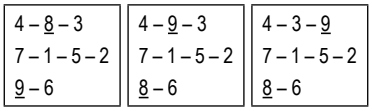

Figura 6. Intercambio Aleatorio (Matriz)

Fuente: elaboración propia.

En la figura 5 se ilustra el intercambio de los trabajos 3 y 2 seleccionados en forma aleatoria, mientras que la figura 6 ilustra el intercambio del trabajo 8 con el trabajo 9 (ambos seleccionados aleatoriamente de las máquinas 1 y 3 respectivamente, las que a su vez son seleccionadas aleatoriamente). Se asume que al aplicar la heurística MV en la máquina 1 la secuencia se reordena, mientras que en la máquina 3 se mantiene. En el proceso de búsqueda, en cada vecindad los intercambios se realizan un número determinado de veces controlado por el parámetro *nIter*.

2.4 *Ant Colony Optimization*

En este trabajo, se utiliza la variante ACO denominada *Ant Colony System* (ACS) de [17] aplicada en forma iterativa a la optimización de la secuencia de trabajos en la máquina que va resultando con mayor *makespan*, por lo que el problema en esta fase se reduce a la sucesiva optimización del *makespan* en problemas de una máquina.

El modelo se aplica a una máquina considerando un grafo completo en el que hormigas artificiales construyen soluciones a través de movimientos aleatorios moviéndose de nodo a nodo (de trabajo a trabajo). Cada arco (i,j) mantiene el nivel τ_{ij} de feromona que depende de la calidad de la solución, lo que sumado a la información heurística η_{ij} ($\eta_{ij} = 1/s_{ijk}$) atrae o desincentiva a las hormigas a utilizar el arco (i,j). El rastro inicial de feromona sobre cada arco (i,j) se define en la forma estándar como $\tau_0 = 1/(n_k \cdot M_k)$, donde n_k es el número de trabajos asignados a la máquina k y M_k el *makespan* de la secuencia inicial entregada por VNS en la máquina k (obtenida mediante la heurística

del mejor vecino). La selección del siguiente nodo para una hormiga se realiza en forma probabilística mediante una regla proporcional aleatoria (ecuación (1)).

$$j = \begin{cases} argmax_{l \in F_i} \left\{ \tau_{il} \left[\eta_{il} \right]^{\beta} \right\}, & siq \le q_0 \\ J, & enotrocaso \end{cases} \tag{1}$$

Estando en el nodo i, una hormiga selecciona el siguiente nodo con probabilidad q_0 de acuerdo con el mayor de los productos $\tau_{il} \cdot \eta_{il}^{\beta}$ (β es un parámetro ACS que relaciona la información actualizada de la red de nodos con la información heurística) sobre los nodos l aún no visitados por la hormiga ($l \in F_i$), y con probabilidad ($1 - q_0$) de acuerdo a la variable aleatoriaJ. Las probabilidades de los valores de J(probabilidad de ir desde el nodo i al nodo j aún no seleccionado por la hormiga) se define según (2):

$$p_{ij} = \frac{\tau_{ij} \eta_{ij}^{\beta}}{\sum_{l \in N_i} \tau_{il} \eta_{il}^{\beta}} \tag{2}$$

En cada iteración cada hormiga pasa de un nodo a otro realizando una actualización local de feromona en el arco (i,j) transitado, según la ecuación (3) utilizando un factor de evaporación ρ, $0 < \rho < 1$:

$$\tau_{ij} \leftarrow (1 - \rho) \tau_{ij} + \rho \tau_0 \tag{3}$$

Al finalizar un ciclo (luego de n_k iteraciones) cada una de lash hormigas ha construido una solución y se realiza una actualización global, utilizando un factor de evaporación γ, $0 < \gamma < 1$, sobre los arcos de la mejor solución conocida:

$$\tau_{ij} \leftarrow (1 - \gamma) \tau_{ij} + \gamma \cdot \frac{1}{c^{bs}} \forall (i, j) \in T^{bs} \tag{4}$$

Donde c^{bs} representa el *makespan* de la mejor solución (secuencia) T^{bs}, encontrada hasta el momento.

3. ESTUDIO EXPERIMENTAL

Para evaluar el método propuesto VNS/ACO se utilizó un conjunto de 90 instancias generadas aleatoriamente por [1] que consideran tiempos de proceso dominantes, donde los tiempos de proceso y *setup* provienen de distribuciones uniformes en los intervalos [125,175] y [50,100] respectivamente. Los tamaños de problema son del tipo $n = 10m$, es decir, en todos los tamaños del problema se consideran en promedio 10 trabajos por máquina, denotándose de la forma m / n. Se consideran 6 tamaños diferentes

(2 / 20, 4 / 40, 6 / 60, 8 / 80, 10 / 100 y 12 / 120) con 15 instancias por cada tamaño. Los datos son los mismos que utilizan [1] con TS, [2] con Meta-RaPS y [3] con ACO, por lo que se sigue esta misma línea investigativa para efectos de comparación del método.

La medida de calidad de la solución se calcula como la diferencia porcentual (ecuación (8)) con respecto a la cota inferiorCI (ecuación 7) calculada como el máximo entre las cotas CI_1 (ecuación 5) y CI_2 (ecuación 6) (ver [18]):

$$CI_2 = max_{j=1,\ldots,n}\left\{min_{i=1,\ldots,n;k=1,\ldots,m}\left[p_{jk}+s_{ijk}\right]\right\} \tag{5}$$

$$CI_2 = max_{j=1,\ldots,n}\ \{min_{i=1,\ldots,n;k=1,\ldots,m}\ [p_{jk}+s_{ijk}]\} \tag{6}$$

$$CI = \max\left(CI_1, CI_2\right) \tag{7}$$

$$DiferenciaPorcentual\left(DP\right) = \frac{C_{max_{método}} - CI}{CI} x100 \tag{8}$$

Para la definición de los parámetros ACO se elige realizar pruebas con los valores recomendados por [19], con los que se generan 36 combinaciones para los parámetros β, ρ y q_0, donde β refleja la importancia de la información heurística. Los valores con los que se realiza la prueba corresponden a β = 2, 3, 4, 5; ρ = 0,1, 02, 0,3 y q_0 = 0,90, 0,95, 0,98, el número de hormigas m se fija en 10, realizando 3.000 ciclos y los factores de evaporación γ y ρ se igualan para reducir el número de parámetros a calibrar. De esta prueba se elige la combinación β = 5, ρ = 0,1 y q_0 = 0,90.

Con respecto a VNS, los tamaños de las vecindades son para la secuencial $(n-1)\cdot n/2$ y para la matricial $\sum_{k=1}^{m-1} n_k \sum_{t=k+1}^{m} n_t$ (con n_k número de trabajos asignados a la máquina k). Si se asume que en promedio se tienen un número n_0 de trabajos en cada máquina el tamaño de la vecindad matricial resulta $n_0^2 \cdot (m-1)\cdot m/2$ lo que para n = 120, m = 12 y n_0 = 10 (problemas de mayor tamaño analizados; los problemas del estudio tienden a tomar soluciones de 10 trabajos por máquina) los tamaños de las vecindades resultan 7.140 y 6.600 para la secuencial y matricial respectivamente.

Se realizan pruebas para determinar el número de iteraciones del algoritmo con valores que oscilan entre 1.000 y 25.000, obteniéndose mejores resultados con 15.000 iteraciones, en orden de magnitud 2 veces el tamaño de la vecindad.

Como ya se explicó, el resultado del algoritmo depende del orden en que se apliquen las vecindades (*swap-s/swap-m* y *swap-m/swap-s*). Se obtienen buenos resultados

para ambas combinaciones de vecindad, así para la comparación con otros métodos de la literatura, se decide extraer el mejor valor de *makespan*, independiente de la combinación de la que provenga.

Las pruebas se realizan utilizando rutinas adaptadas de SPS_Optimizer [20] y se ejecutan en un computador con sistema operativo de 32 bits *Windows 7 Profesional*, con procesador Dual Core AMD de 1.65 GHz y 2.0 GB de memoria RAM.

4. RESULTADOS

Se realizan pruebas para determinar la efectividad del algoritmo VNS/ACO frente al algoritmo VNS. Su comportamiento se muestra en la figura 7, donde se encuentran los tamaños de problema en el eje horizontal y la diferencia porcentual con respecto a la cota inferior en el eje vertical. Es importante mencionar que en los problemas pequeños (2, 4 y 6 máquinas) a medida que aumenta el tamaño del problema se acorta la brecha entre la diferencia porcentual promedio de ambos métodos. En el caso de los problemas de mayor tamaño (8, 10 y 12 máquinas) no se observa un patrón que describa el comportamiento. Para comparar con otros métodos de la literatura se elige utilizar sólo los resultados obtenidos con VNS/ACO, el resumen se presenta en la tabla 4.

Tabla 4. Diferencia porcentual promedio con respecto a la cota inferior

DP	2 / 20	4 / 40	6 / 60	8 / 80	10 / 100	12 / 120
Meta-RaPS[2]	2,60	3,88	4,37	4,62	5,22	4,25
TabuSearch[1]	3,97	5,14	6,38	6,33	6,79	7,05
ACO II [3]	2,52	3,44	4,58	4,44	4,91	4,64
VNS	2,99	3,55	4,27	4,54	4,44	4,77
VNS/ACO	2,70	3,43	4,17	4,46	4,41	4,67

Fuente: elaboración propia.

Con respecto a los otros métodos, VNS/ACO ofrece la menor diferencia porcentual promedio en 3 de los 6 tamaños de problema estudiados (4, 6 y 10 máquinas) con diferencias porcentuales de 3,43 %, 4,17 % y 4,41 %. También en las configuraciones de 2 y 8 máquinas ofrece la segunda menor diferencia porcentual promedio con un 2,70 % y 4,46 % respectivamente. Con respecto a la cantidad de mejores soluciones por tamaño de problema (ver tabla 5), ofrece al menos una mejor solución en todos los tamaños, con un mínimo de 1 mejor solución en el tamaño 12 / 120 (6,67 %) y un máximo de 13 mejores soluciones en la configuración 10 m / 100 n. Sólo ACO entrega también una mejor solución en todos los tamaños y presenta las menores diferencias porcentuales en 2 tamaños (2 / 20 y 8 / 80) con un 2,52 % y 4,44 % respectivamente.

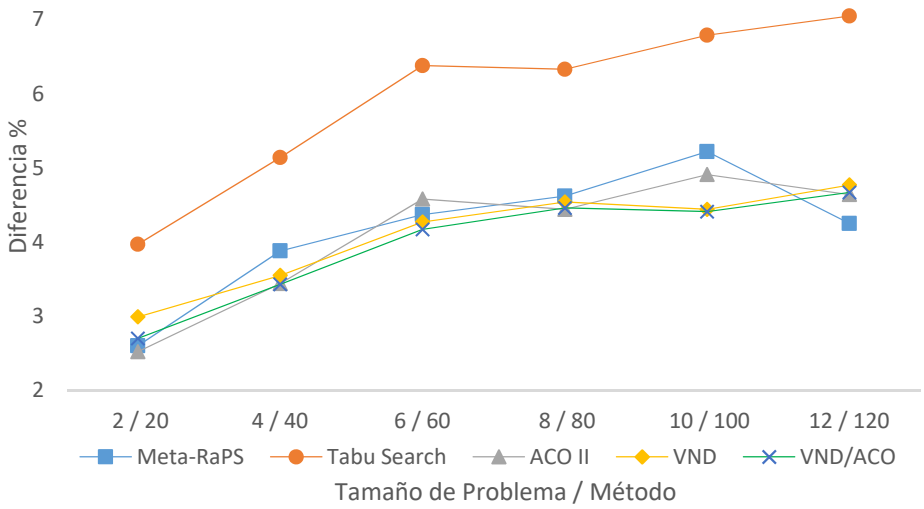

Figura 7. Diferencia porcentual promedio con respecto a la cota inferior

Fuente: elaboración propia.

Tabla 5. Porcentaje de mejores soluciones

Tamaño	Meta-RaPS	TS	ACO II	VNS/ACO
2 / 20	60,00	0,00	86,67	33,33
4 / 40	0,00	0,00	60,00	46,67
6 / 60	20,00	0,00	6,67	73,33
8 / 80	13,33	0,00	66,67	40,00
10 / 100	0,00	0,00	13,33	86,67
12 / 120	73,33	0,00	20,00	6,67

Fuente: elaboración propia.

5. CONCLUSIONES

El problema de máquinas paralelas no relacionadas con tiempos de *setup* dependientes de la secuencia se resuelve a través de un algoritmo VNS hibridizado secuencialmente con la metaheurística ACO en su versión ACS, la que reordena sucesivamente los trabajos en la máquina de mayor *makespan* de la solución entregada por VNS en su versión VND.

Se comprueba la efectividad de la hibridización VNS/ACO ya que, al considerar la desviación porcentual media con respecto de una cota inferior del *makespan,* se observa una mejora de la solución VNS en el conjunto de instancias evaluado. Al compararse con otros métodos de la literatura, se observa que en el 50 % de los tamaños de problema estudiados presenta la menor desviación porcentual promedio respecto

de la cota inferior calculada, además de mostrar un buen desempeño con respecto al porcentaje de mejores soluciones encontradas, lo que lo hace un algoritmo competitivo.

En virtud de los resultados obtenidos, como trabajo futuro se proyecta continuar el trabajo en dos direcciones, por un lado, conducir experimentos de análisis de varianza considerando bancos de problemas generados aleatoriamente y, por otro lado, realizar aplicaciones del método a problemas reales.

REFERENCIAS

[1] M. Helal, G. Rabadi y A. Al-Salem, "A tabu search algorithm to minimize the makespan for the unrelated parallel machines scheduling problem with setup times," *International Journal of Operations Research*, vol. 3, n.° 3, pp. 182–192, 2006.

[2] G. Rabadi, R. J. Moraga. y A. Al-Salem, "Heuristics for the unrelated parallel machine scheduling problem with setup times",*Journal of Intelligent Manufacturing*, vol. 17, n.° 1, pp. 85–97, 2006.

[3] J-P. Arnaout, R. Musa y G. Rabadi, "A Two-stage Ant Colony Optimization Algorithm to Minimize the Makespan on Unrelated Parallel Machines—part II: Enhancements and Experimentations," *Journal of Intelligent Manufacturing*, vol. 25, n.° 1, pp. 43–53, 2014.

[4] E. Vallada y R. Ruiz, "A Genetic Algorithm for the Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times," *European Journal of Operational Research*, vol. 211, n.°3, pp. 612–622, 2011.

[5] K-Ch. Ying, Z-J. Lee y Sh-W. Lin, "Makespan minimization for scheduling unrelated parallel machines with setup times," *Journal of Intelligent Manufacturing*, vol 23, n.° 5, pp. 1795–1803, 2012.

[6] J. Behnamian, M. Zandieh y S. F. Ghomi, "Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm," *Expert Systems with Applications*, vol. 36, n.° 6, pp. 9637–9644, 2009.

[7] M. Mateo, J. Teghem y D. Tuyttens, "A bi-objective parallel machine problem with eligibility, release dates and delivery times of the jobs," *International Journal of Production Research*, vol 56, n.° 3, pp. 1030–1053, 2018.

[8] Y-I. Kim y H-J. Kim, "Rescheduling of unrelated parallel machines with job-dependent setup times under forecasted machine breakdown," *International Journal of Production Research*, Online: junio 2020. https://doi.org/10.1080/00207543.2020.1775910

[9] N. Mladenovic y P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, n.° 11, pp. 1097-1100, 1997.

[10] R. Driessel y L. Mönch, "Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times," *Computers & Industrial Engineering*, vol. 61, n.° 2, pp. 33–345, 2011.

[11] P. Hansen, N. Mladenovic y J. Moreno, "Búsqueda de entorno variable," *Revista Iberoamericana de Inteligencia Artificial*, vol. 7, n.°19, pp. 77–92, 2003.

[12] M. Dorigo, "Optimization, Learning and Natural Algorithms (in Italian)." Ph.D. Thesis, Dipartimento de Elettronica, Politécnico de Milán (Italy),1992.

[13] E. Salazar y C. Avila, "Heurística GRASP para la minimización del makespan en máquinas paralelas no relacionadas con tiempos de preparación dependientes de la secuencia," *Ingeniare – Revista Chilena de Ingeniería*, vol. 25, n.°3, pp. 524 - 34. 2017.

[14] T. Peñaloza, "Algoritmo híbrido GRASP/ACO aplicado al problema de minimización de makespan en máquinas paralelas no relacionadas con tiempos de *setup* dependientes de la secuencia". Memoria de Título Ingeniería Civil Industrial, Universidad de Concepción (Chile), Julio 2015.

[15] E. Salazar y J. C. Medina, "Minimización del makespan en máquinas paralelas idénticas con tiempos de preparación dependientes de la secuencia utilizando un algoritmo genético," *Ingeniería, Investigación y Tecnología*, vol. 14, n.° 1, pp. 43-51, 2013.

[16] O. H. Ibarra y C. E. Kim, "Heuristic algorithms for scheduling independent tasks on non-identical processors," *Journal of the ACM*, vol. 24, n.° 2, pp. 280-289, 1977.

[17] M. Dorigo y L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, n.° 1, pp. 53-66, 1997.

[18] J-P. Arnaout, G. Rabadi y R. Musa, "A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times," *Journal of Intelligent Manufacturing*, vol. 21, n.° 6, pp. 693-701, 2009.

[19] M. Dorigo y T. Stützle, *Ant colony optimization*, MIT Press, 2004.

[20] E. Salazar, "Programación de Sistemas de Producción con SPS_Optimizer," *Revista ICHIO*, vol. 1, n.°2, pp. 33-46, 2010.